Cooperative Announcement-based Caching for Video-on-Demand Streaming

Maxim Claeys, *Student Member, IEEE*, Niels Bouten, *Student Member, IEEE*, Danny De Vleeschauwer, Werner Van Leekwijck, Steven Latré, *Member, IEEE*, and Filip De Turck, *Senior Member, IEEE*

***Abstract*—Recently, Video-on-Demand (VoD) streaming services like Netflix and Hulu have gained a lot of popularity. This has led to a strong increase in bandwidth capacity requirements in the network. To reduce this network load, the design of appropriate caching strategies is of utmost importance. Based on the fact that, typically, a video stream is temporally segmented into smaller chunks that can be accessed and decoded independently, cache replacement strategies have been developed that take advantage of this temporal structure in the video. In this paper, two caching strategies are proposed that additionally take advantage of the phenomenon of binge watching, where users stream multiple consecutive episodes of the same series, reported by recent user behavior studies to become the everyday behavior. Taking into account this information allows us to predict future segment requests, even before the video playout has started. Two strategies are proposed, both with a different level of coordination between the caches in the network. Using a VoD request trace based on binge watching user characteristics, the presented algorithms have been thoroughly evaluated in multiple network topologies with different characteristics, showing their general applicability. It was shown that in a realistic scenario, the proposed election-based caching strategy can outperform the state-of-the-art by 20% in terms of cache hit ratio while using 4% less network bandwidth.**

***Index Terms*—Cooperative caching, Multimedia streaming, Video-on-Demand, Binge watching**

I. Introduction

Traditionally, channel-based networks (e.g., satellite, cable networks) were used to distribute linear TV, benefiting from the broadcast nature of these networks. However, while in linear TV the amount of video traffic is proportional to the number of channels, in an on-demand system the traffic is proportional to the number of users. This leads to a major increase in capacity requirements for on-demand video delivery on the backbone. By deploying intermediary caches, closer to the end-users, popular content can be served faster and without increasing backbone traffic. Since video interests are shared between users and their viewing interests overlap in time, deploying these caches can substantially reduce the required capacity by taking advantage of these overlapping sessions. Caching networks (e.g., Content Delivery Networks (CDNs)) were traditionally used to deliver web content in a scalable way and reduce latency by temporally storing part of the content in a network of caches close to the end-users. However, the use of caching in a streaming environment differs from the traditional web-based caching, since the objects are typically much larger and are to be consumed directly after they have been requested. Therefore, video delivered over caching networks is typically temporally segmented into smaller chunks that can be accessed and decoded independently.

Designing an appropriate replacement strategy for such caching networks is of utmost importance to achieve high caching efficiency and reduce the network load. Most caching algorithms are based on observations and take into account the recency or frequency of requests to calculate the rank of each of the cached objects. When caching temporally segmented video, as is the case in HTTP-based streaming systems, the caching algorithm can also take into account the temporal structure of the video. That is, when the streaming application requests the n-th segment at a specific point in time, the caching strategy can take advantage of the knowledge that with high likelihood the (n+1)-th and subsequent segments will be consumed shortly after this segment. Therefore, when multiple streaming sessions request the same segmented content, the caching strategy can take advantage of this knowledge to keep the segments in cache that are to be consumed in the near future.

Caching strategies can be further improved by taking into account additional information, such as content popularity and regional differences. The optimal caching algorithm for an isolated cache (known as Belady's MIN [1]) takes advantage of the knowledge on future requests to discard the objects for which the next request occurs the furthest in the future. In a real system, this information on future requests is not directly available. However, studies[1] on user behavior for over the top (OTT) streaming services report that respectively 88% and 70% of the Netflix and Hulu Plus users stream three or more consecutive episodes of the same TV show (referred to as *binge watching*). Furthermore, binge watching is reported to become the everyday behavior with users streaming on average 2.32 episodes per sitting, resulting in about 57% of

Manuscript received November 06, 2015.

M. Claeys and N. Bouten are funded by a grant of the Agency for Innovation by Science and Technology in Flanders (IWT). The research was performed partially within the ICON SHIFT-TV project (under grant agreement no. 140684). This work was partly funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

M. Claeys, N. Bouten, and F. De Turck are with the Department of Information Technology, Ghent University - iMinds, Ghent, Belgium (email: maxim.claeys@intec.ugent.be).

D. De Vleeschauwer, and W. Van Leekwijck are with Nokia Bell Labs, Antwerp, Belgium.

S. Latré, M. Claeys and N. Bouten are with the Department of Mathematics and Computer Science, University of Antwerp - iMinds, Antwerp, Belgium.

Digital Object Identifier XX.XXXX/XXXX.XXXX.XXXXXXX

[1]Nielsen - http://www.nielsen.com/us/en/insights/news/2013/binging-is-the-new-viewing-for-over-the-top-streamers.html

the streaming sessions that could be announced in advance [2]. Taking into account these announcements allows us to predict future segment requests and thus approximating the theoretical optimal caching strategy. Furthermore, Service Providers (SPs) could give incentives (e.g., discounts, higher resolution video (4K)) to users when they announce their streaming sessions in advance.

In this paper, a novel caching strategy is proposed that takes advantage of the inferred knowledge of future segment requests when streaming segmented video and additionally exploits the added knowledge when streaming clients announce the videos that will be streamed in the near future. Using this additional information, we are able to estimate the future reuse times of the segmented content. This allows us to increase the caching efficiency compared to using only reuse time predictions based on the structure of the segmented video. Two caching strategies are proposed, each applying a different type of cooperation inside the caching network. The first approach, based on the strategy presented in our previous work [3], applies a simple threshold-based cascading strategy to provide a basic level of coordination. The second proposal elaborates on this approach by adding a more advanced election-based coordination strategy. Furthermore, compared to our previous work, experiments have been performed using different network topologies and request traces, based on recent analysis of user behavior in a Video-on-Demand (VoD) scenario.

The main contributions of this paper are threefold. First, we propose a novel election-based caching strategy that outperforms the current state-of-the-art strategies by including knowledge on segmented video streaming and announced future video requests. Second, we use user behavior models to emulate video interruptions and evaluate their impact on the proposed and state-of-the-art caching strategies. Finally, the proposed strategy is thoroughly evaluated in a distributed caching scenario on multiple topologies.

II. Related Work

A. Cache Replacement Strategies

Web caching was introduced as a way to save traffic and minimize the perceived delay on the Internet by storing data closer to the end-user. Many cache replacement algorithms have been proposed in the past and stem from replacement strategies for both web proxy caching and CPU caching. Least Recently Used (LRU) is based on the recency of the cached objects and discards the least recently used items first. This type of caching algorithms is known to attach too much importance to unpopular objects, awarding them the highest rank upon a single request. Least Frequently Used (LFU) determines the number of requests for an object over a period of time and discards the least frequently used objects first. This type of caching algorithms is known to attach as much importance to ancient as to recent history. To account for this, dynamic aging factors are introduced to improve the performance in LFU Dynamic Aging (LFU-DA). Adaptive Replacement Cache (ARC) balances between LRU and LFU by using more complex algorithms to determine the rank [4].

The aforementioned replacement strategies are widely used in web caches. Caching for video streaming differs from caching web content in a number of ways. First, video objects are typically much larger than traditional web objects. Second, there is a difference in popularity evolution for streaming videos compared to web content. Furthermore, in contrast to web objects, having only the beginning of a video allows a video application to already start playout while the download continues [5]. This has led to the adoption of segment-based caching of multimedia streams. These segments can have variable size, e.g., smaller segments at the start of the video to decrease start-up delays. *Wu et al.* define three segmentation approaches for proxy caching: fixed length segments, pyramid with exponentially increasing segment sizes and skyscraper where sizes increase slower compared to the pyramid scheme by repeating each layer n times [6], [7]. The authors conclude that segmentation approaches are particularly effective when the cache size is limited, media popularity changes over time, requests are spread over a large number of media objects and when the media file size and library is large.

Chen et al. propose a streaming proxy system called *Hyper Proxy*, which uses active prefetching for in-time delivery of uncached segments to address the proxy jitter problem [8]. This work was extended by adding segmentation strategies based on object popularity and by using predictions of future segment requests to preload uncached segments [9]. In previous work, a proactive cache management system for multi-tenant CDNs was proposed that optimizes content placement and server selection based on content popularity and geographical distribution of requests [10]. In this paper we focus on reactive caching approaches leveraging video structure and request announcements rather than proactively placing content into network caches. Other approaches use a two-tier cache that distinguishes between to-be-played and possibly played segments, partitioning the cache in two layers that are dynamically scaled [11]. The approach of dynamic cache partitioning has also been applied by *Wauters et al.*, where caching decisions in a time-shifted television service are based on content popularity metrics [12]. Popularity-based caching approaches have also been proposed by *De Vleeschauwer et al.* [13].

Marinca et al. use an analysis of the clients' request over passed time intervals to predict the content that will be requested in the near future [14]. By using a history window of the past 24 hours allows reducing the traffic with about 30%. Others evaluate the impact of dynamic sizing between prefix and suffix caching for segmented video [15]. *Hong et al.* propose a ranking scheme that follows the dynamicity of the video library [16]. The rank is based on the linearity of the video that if a chunk is requested, the next chunk will be requested with high probability in the near future. To this end each segment is assigned a value based on the number of viewers who are watching the video and will probably request the segment in the future. This value reflects the number of hits that this particular segment will receive in the future. Based on this value a rank is calculated that evicts the segments with the least chance of being reused in the future. In this paper, we additionally take into account the timing information of future requests by considering video request announcements to further increase the cache performance.

The MIN cache replacement algorithm proposed by *Belady et al.* is an offline eviction strategy that has been proven to be optimal [1]. It requires advanced knowledge of the requested content, and based on this information, chooses to evict the item in the cache whose next request occurs furthest in the future. *Van Roy et al.* propose a proof of optimality for the MIN cache replacement algorithm based on a dynamic programming argument [17]. *Wu et al.* make use of the natural linear time structure of segmented video to propose a caching algorithm based on the exact reuse time [18]. This reuse time indicates when a cached segment will be reused and allows the eviction strategy to determine the segment request that will occur the furthest in the future. The authors do not take into account the delivery times between the different nodes in the network, only a single cache is assumed and the sessions are never interrupted. Our proposed approach not only uses the temporal structure of segmented video but also includes information of announced sessions. Furthermore, we also use interruption models to simulate user churn and use a hierarchy of caches where the delivery times between the different nodes are taken into account.

Recently, protocols have been proposed that support the announcement of deadlines, applied in this work. Shared Content Addressing Protocol (SCAP) is a transport protocol aimed at optimizing the delivery process that allows in-network intermediary elements to participate in delivering the content [19]. Furthermore it offers support for intermediary caches, multicast-like delivery of shared content requests and announcing deadlines for the delivery of the content. Similar features are also provided by Information-Centric Networking (ICN) architectures [20]. These architectures allow content to be duplicated everywhere in the network, enabling in-network caching at a fine granularity [21].

B. Cache Coordination Strategies

In traditional hierarchical web caching systems there is no cooperation between the different nodes which deploy a replacement algorithm locally (e.g., LRU). Unsatisfied requests are forwarded up the hierarchy, resulting in copies of the requested objects being placed in every cache along the return path to the requester. This data replication leads to a poor utilization of the global cache storage capacity. *Che et al.* propose a cooperative hierarchical caching strategy where caching an object in a certain node is based on the access frequency for the document at this level [22], [23]. Furthermore, a document that is replaced at a certain level is further cached at the upper level cache, if not already. This ensures both that popular items reside closer to the end user, while avoiding that documents with decreasing popularity are completely removed from the cache hierarchy.

Tang et al. investigated cache management strategies for en-route web caching and proposed a caching scheme that integrates both object placement and replacement policies [24]. In the proposed scheme, the cache status information along the routing path of a request is used to dynamically determine where to cache the object and which objects to replace. *Jiang et al.* extend upon this work and argue that previous solutions only solved restricted partial problems. The authors propose a dynamic programming approach which computes the global optimal solution with and without prefetching [25]. Previous work on hierarchical and en-route caching focused on web caching, which is substantially different compared to the caching of multimedia objects. In more recent work, *Poularakis et al.* focus on the delivery of VoD content in hierarchical cache topologies such as Internet Protocol television (IPTV) networks tend to have [26]. A heuristic approach is taken in which items are iteratively replaced by other items at the caching nodes if this yields a reduction in access cost, which is impacted by both latency and bandwidth consumption. The authors assume unit size objects which allows them to calculate an optimal solution to the content placement problem in polynomial time.

Also in CDN research, cooperative caching has been a subject of interest over the past years. *Ni et al.* propose a hierarchical overlay architecture for CDNs where different servers cooperate through intracluster and intercluster cooperative caching schemes [27]. For intracluster cooperation a portion of each cache is assigned to cooperate in a hashing based system, while for intercluster cooperation a query-based scheme is proposed where each cluster representative queries its neighboring clusters for missing content. The time-shifted and VoD nature of new and emerging video services defies the broadcast paradigm of the underlying conventional TV networks. To cope with the increased bandwidth demands these services incur, *Borst et al.* formulate the content placement problem as a linear program in order to benchmark the globally optimal performance [28]. A lightweight cooperative content placement algorithm is proposed in which each nodes modifies its local content store to achieve the maximum gain in global utility. *Zhang et al.* performed a survey on cooperative caching in CDNs and conclude that increased cooperation between caches is required to decrease the bandwidth requirements and caching redundancy [29]. This is also confirmed by *Li et al.* who propose a genetic algorithm implemented on the MapReduce framework for the placement of video chunks [30]. The authors achieve similar hit ratios as with LRU, but are able to reduce the workload on the congested links.

In ICN, one of the simplest non-cooperative caching strategies is called Leave Copy Everywhere (LCE), which copies the object at each node along the downloading path. In a recent survey, different strategies for both explicit and implicit cache coordination are discussed [31]. *Li et al.* propose a cooperative caching strategy which aims at decreasing the cross-domain traffic for on-demand video streaming. They propose a hashing-based neighborhood caching strategy in which each node only stores chunks of which the hash of the id matches their assigned range [32]. Another proposed approach is to use age-based caching algorithms where replicas closer to the edge and with higher popularity are assigned with a higher age [33]. The routers adjust the contents age field as it travels along the path. The paper claims that it is straightforward to use the scheme for chunk-level caching, however no measures are proposed to take advantage of the temporal structure of these chunks. *Sourlas et al.* propose to put distributed autonomic managers in the cache-enabled nodes which decide on the

information items to cache. The authors propose different levels of autonomicity, each with a different degree of communication overhead [34]. *Li et al.* propose an optimal strategy for provisioning storage capacity which optimizes the overall network performance and cost [35]. A routing-aware content placement algorithm is proposes which runs on a centralized server and assigns content to store to each Content-Centric Networking (CCN) router. There exists a trade-off between the content coordination cost (provisioning storage) and network routing performance in CCN. In non-coordinated caching, less distinct contents are stored and reduce the efficiency of the caching "cloud".

III. Client messaging behavior

In the approaches that will be presented in the next section, the caching nodes keep track of the start times of video streaming sessions. To this end, the caches rely on messaging from the video clients. The clients can send information in the form of three types of messages: (i) session announcements, (ii) session initiation notifications and (iii) session finishing/canceling notifications. In this section, the messaging behavior of the video client is described. The client sends each message to the first cache on the path to the content server. How the messaging is handled inside the network depends on the caching algorithm and will be described in Section IV. To enhance the clarity of the remainder of this work, it should be emphasized that a *streaming session* is referred to as consuming a single video while a *sitting* consists of multiple consecutive streaming sessions of the same user.

A. Session announcements

As described earlier, users often watch multiple consecutive episodes of the same series in a single sitting. This phenomenon is often referred to as binge watching. Therefore, when a user watches an episode of a specific series, there is a significant probability that the user will watch the next episode of the same series once the current episode finishes. The client can use this information to generate a session announcement. When the client starts streaming episode i of a specific series at time t_i, the predicted start time of episode $i+1$ can be calculated as $t_{i+1} = t_i + d_i$, where d_i denotes the duration of episode i. The announcement for episode $i+1$ contains the video ID of the next episode and its estimated start time t_{i+1}. This announcement message can be sent into the caching network anywhere between time t_i and t_{i+1}, based on the relative announcement delay parameter $\beta \in [0; 1]$. Concretely, the announcement will be send at time $a_{i+1} = t_i + \beta \times d_i$. It is important to note that no prior announcements are made for the first episode of a sitting. Furthermore, for episodic content, this strategy introduces exactly one false announcement per sitting (i.e., the announcement made for the episode after the current one when the current one is abandoned). For movies, no announcements are made.

B. Session initiations

When the playout of a video is initiated, the client informs the caching network about the video that is started and the actual start time of the streaming session. It is important to note that this messaging could also be performed implicitly by interpreting the first segment request of a specific video. For clarity reasons however, explicit messaging of session initiation is assumed in the remainder of this work.

C. Session expiration

At the end of a streaming session, the video client informs the caching network that the current video playout is finished. When an announcement was made earlier for the next episode of a series being watched, this announcement can optionally be canceled. The message used to finish a streaming session or to cancel an announcement contains the ID of the finished or canceled video and the (announced) start time of the video.

IV. Session-aware cache replacement

The knowledge about start times of video streaming sessions can be used by the caching algorithms to infer the future request times of individual video segments. Based on this information, replacement decisions can be made. In this work, two different caching strategies are proposed. First, Section IV-A describes an announcement-based caching strategy with a basic form of communication in the caching network, as proposed in our previous work [3]. Next, additional coordination between the caches is added in the form of an election strategy, as will be described in Section IV-B. For both caching strategies, the message handling process and the segment replacement strategy will be discussed.

The goal of both coordination strategies is to find a node in the caching network that will take the responsibility of caching the video for each streaming session. In the remainder of this paper, a node taking responsibility of caching a specific video will be denoted as that node *accepting* the streaming session.

A. Threshold-based caching strategy

As described in Section III, streaming session can either be announced prior to playout or be advertised at the moment the session starts. In the threshold-based caching strategy, sessions that were announced in advance play a different role in the replacement strategy than sessions that were only advertised at their start. Therefore, each caching node keeps track of two sets of streaming sessions. The set of announced sessions Ann will be maintained in such a way that it contains all announced sessions that were accepted by the local caching node. These sessions can either be active sessions or future sessions. Similarly, the set of perceived sessions Per will at any time contain all active streaming sessions that the cache is aware of. These sessions can either have been announced but not accepted locally or not have been announced at all. For Per, no future sessions are considered. For each session in Ann and Per, the streamed video v and its start time t_v is recorded.

1) Message handling: To keep track of the session information, a basic form of communication exists between the different caches in the network. This communication is implemented as a threshold-based cascading strategy.

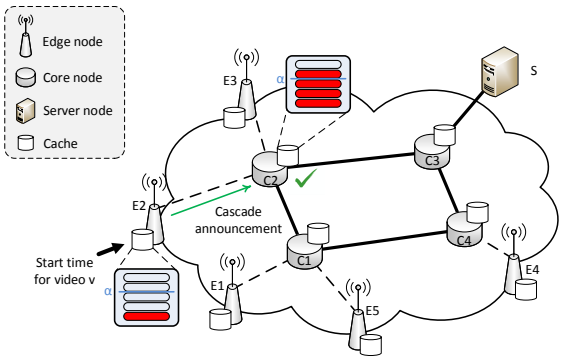


Fig. 1. Illustration of the session announcement handling process in the threshold-based caching strategy.

a) Session announcements: When a caching node receives a session announcement message, it decides autonomously whether to accept the session locally or to cascade it to the next hop on the path to the content server. A node will only decide to accept the announced session, and add it to the set of accepted announced sessions Ann, if it estimates to be able to serve at least a relative part α of the video from its local cache. This estimation is based on the number of segments of the considered video that are already present in the cache, combined with the segments that are estimated to arrive before the new session starts, based on known earlier sessions for the same video. The value of α serves as a session acceptance threshold and provides a basic form of coordination between the caches. If the node is not able to meet the threshold α, it does not accept the session and cascades the announcement to the next hop. This cascading process continues until either the announcement is accepted or the last hop on the path to the server is reached. An illustration of this process is given in Figure 1. A client sends a session announcement for video v to the edge node $E2$. As the number of cached segments for video v is below the session acceptance threshold α, $E2$ cascades the announcement to the next hop $C2$ where it is accepted and the cascading process is terminated.

b) Session initiations: Upon arrival of a session initiation message for video v, the cache checks if it already has an accepted announced session for that video with the current time as start time (i.e., if Ann already contains a start time t for video v). If this is not the case, the session information is added to the set of perceived sessions Per. In this way, Ann and Per are fully distinct. However, it is important to note that even though Ann and Per are fully distinct, both sets can contain different start times t_v for the same video v. Furthermore, a single set can contain multiple start times for the same video. Next, the session initiation message is sent to the next hop on the path to the server until the last hop is reached.

c) Session expiration: When a streaming session ends, the session information can be removed from the caching nodes. When a cache receives a session expiration message for an announced start time t of video v, it removes the session from the set of accepted announced sessions Ann. If the session was not accepted locally, the expiration message is cascaded to the next hop. Otherwise, the cascading process is finished. Perceived sessions $s_v \in Per$ are only removed when the session is known to have ended, even if it was interrupted prematurely. For a session s_v with start time t_v, this is the case when $t > t_v + L_v$, with t and L_v denoting the current time and the total length of video v respectively.

2) Replacement strategy: Using the information contained in the start times of the announced and perceived sessions, each node can decide which segments to keep in its cache, and which can be removed. For this purpose, the concept of earliest reuse times is applied. For each start time t_v for a specific video v, the reuse time $r_{v_x}^{t_v}$ of each segment v_x of that video can easily be calculated using (1), where x and l_v respectively denote the sequence number of segment v_x and the segment length for video v. If $r_{v_x}^{t_v} < t$, with t denoting the current time, the reuse time is set to infinity. The earliest announced reuse time $e_{v_x}^A$ is then obtained as the lowest reuse time $r_{v_x}^{t_v}$ of all start times t_v of accepted announced sessions Ann for video v, as expressed in (2). Similarly, the earliest perceived reuse time is obtained using (3). When there are no accepted announced or perceived sessions for video v, respectively $e_{v_x}^A$ or $e_{v_x}^P$ are said to be equal to infinity for every segment v_x of that video.

$$r_{v_x}^{t_v} = t_v + x \times l_v \tag{1}$$

$$e_{v_x}^A = \min_{t_v \in Ann} r_{v_x}^{t_v} \tag{2}$$

$$e_{v_x}^P = \min_{t_v \in Per} r_{v_x}^{t_v} \tag{3}$$

When a segment v_x of video v arrives at a node, the node has to decide whether or not to add it to its local cache C, if it is not already cached. If the remaining available caching capacity is insufficient to store the new segment v_x, another segment can be selected for removal. This procedure is outlined in Algorithm 1. First, the earliest announced and perceived reuse times $e_{v_x}^A$ and $e_{v_x}^P$ of the newly arrived segment are calculated as described above (line 1). Next, a segment $s' \in C \cup \{v_x\}$ is selected as a candidate for eviction. This segment s' is selected as the segment with the maximal earliest announced reuse time: $s' = \arg\max_{v_i' \in C \cup \{v_x\}} e_{v_i'}^A$ (lines 4-6). It is important to keep in mind at this point that when there are no accepted sessions for the video v' of a segment v_i', the earliest announced reuse time $e_{v_i'}^A$ is equal to infinity. In this way, when there are cached segments for which no session was accepted locally, these are always preferred for eviction. When multiple segments with the same maximal earliest announced reuse time exist, the segment with the maximal earliest perceived reuse time is selected: $s' = \arg\max_{v_i' \in C \cup \{v_x\}} e_{v_i'}^P$ (lines 7-9). The LRU order is used as a final tiebreaker (lines 10-13). When the evicted segment s' is a cached segment ($s' \in C$), it is removed from the cache and replaced by the new segment v_x (lines 18-20).

The rationale behind this approach is to keep the segments in the cache that will be needed in the nearest future. However, accepted announced sessions are always prioritized, even if another segment has an earlier perceived reuse time. This is done to ensure that the cache will not violate the intent expressed

Algorithm 1 Outline of the eviction strategy of the threshold-based caching approach on arrival of a new segment v_x.

```
Calculate e^A_{v_x} and e^P_{v_x}
s' ← v_x
for v'_i ∈ C do
    Calculate e^A_{v'_i} and e^P_{v'_i}
    if e^A_{v'_i} > e^A_{s'} then
        s' ← v'_i
    else if e^A_{v'_i} = e^A_{s'} then
        if e^P_{v'_i} > e^P_{s'} then
            s' ← v'_i
        else if e^P_{v'_i} = e^P_{s'} then
            Calculate LRU ranks LRU_{v'_i} and LRU_{s'}
            if LRU_{v'_i} > LRU_{s'} then
                s' ← v'_i
            end if
        end if
    end if
end for
if s' ≠ v_x then
    Remove s' from C
    Add v_x to C
end if
```

when accepting the sessions and ending the cascading chain. It is important to note that when no sessions are announced, the algorithm behaves as a purely reuse time-based replacement strategy without coordination between the caches, comparable to the approach proposed by *Wu et al.* [18], only taking into account the temporal structure of videos in the form of earliest perceived reuse times.

B. Election-based caching strategy

To further increase the performance of the caching approach, the influence of the level of additional coordination between the different caches is investigated. The goal of this election-based caching approach is to select the optimal location to cache a specific video and avoid unnecessary duplicates on the shared network path. In this caching strategy, each caching node keeps track of both a set of known sessions Kno and a set of accepted sessions Acc. In contrast to the threshold-based caching strategy, Kno will contain both active sessions and announced future sessions and Acc will contain accepted sessions for both sessions that were announced and that were only reported at the start.

1) Message handling: To achieve coordination in the caching network, an election-based strategy has been applied.

a) Session announcements: When the video client sends a session announcement message to the first cache, an election process is initiated between all caches along the path to the server. When a caching node receives the announcement, the session is added to the set of known sessions Kno. Next, the cache calculates the estimated increase in number of cache hits that would be achieved when the announced video would be cached locally. This calculation takes into account the segments that are already present in the cache and the sessions that are known up to know. To clarify this calculation, assume there is an announcement at time t for streaming video v, consisting of 100 segments of 1 second, with start time $t_x > t$. After adding this session to the set of known sessions, Kno contains 3 sessions for video v, with start times t_x (the newly announced session), $t_y = t - 30$ (a streaming session that has already started) and $t_z > t$ (a future session that was announced earlier). At the moment, suppose that 10 segments of video v are present in the cache. Given the session information, one can estimate that 270 segment requests for v will arrive in the future: as $t_x > t$ and $t_z > t$, two sessions still have to start while 30 segments have already been streamed in the session with start time t_y. As the cache only contains 10 out of 100 segments at the moment, 90 segments will still have to be cached, resulting in 90 cache misses. In total, we can estimate 180 future cache hits when video v would be cached.

Similarly, when there is not sufficient available capacity to store the entire video, the minimum amount of lost cache hits due to the removal of cached content is calculated. By subtracting this hit loss from the estimated additional future hits, the expected local gain can be calculated. Next, this potential local gain is passed to the next hop on the path to the server, where the same process is performed. When the last hop on the path is reached, the node with the highest potential gain is selected as the winner of the election process. When none of the caches has a positive potential gain, the election has no winner. The result of the election is cascaded back along the reverse path. When the result arrives at the winner, the announced session is added to the set of accepted sessions Acc. When segments of a video v' for which the node had accepted a session have to be removed from the cache to be able to accommodate for the newly announced session, all sessions for v' are removed from Acc. In this way, the caching capacity is guaranteed to be always sufficient to store all segments of the videos for which sessions are accepted in Acc. An illustration of the election process is given in Figure 2. A client sends a session announcement for video v to the edge node $E2$. The potential gain δ is calculated locally and forwarded to the next hop $C2$. There, the potential gain ϵ is calculated and forwarded to the last hop $C3$ as the gain is higher that the previous maximal gain ($\epsilon > \delta$). At node $C3$, the potential local gain is calculated as ζ, which is lower that the previous gain ϵ. As $C2$ has the highest potential gain, $C2$ is selected as the winner of the election and the result is cascaded back to the first node $E2$.

b) Session initiations: Upon arrival of a session initiation message for video v, the caches behave similarly as for session announcements. In this case, an election is started for an implicit announcement for video v with start time t being the current time. The details of the election process are described above.

c) Session expiration: When a streaming session ends, the session information can be removed from the caching nodes. When a cache receives a session expiration message for a streaming session of video v with start time t, the session information is removed from both the set of known sessions Kno and the set of accepted sessions Acc.

2) Replacement strategy: For the segment replacement strategy, a reuse time-based algorithm is applied, similar to the one presented earlier for the threshold-based strategy. How-

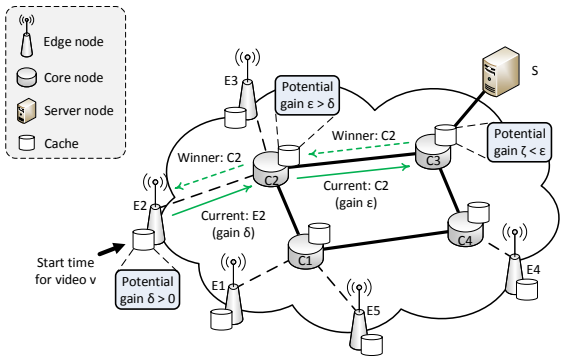


Fig. 2. Illustration of the session announcement handling process in the election-based caching strategy.

ever, as reuse times are calculated based on all known sessions, no distinction is made between announced or perceived reuse times. The earliest reuse time e_{v_x} of a segment v_x of video v is obtained as the lowest reuse time $r_{v_x}^{t_v}$ of all start times t_v of known sessions Kno for video v, as expressed in (4). When there are no known sessions for video v, e_{v_x} is said to be equal to infinity for every segment v_x of that video.

$$e_{v_x} = \min_{t_v \in Kno} r_{v_x}^{t_v} \tag{4}$$

The cache replacement procedure is outlined in Algorithm 2. First, the earliest reuse time e_{v_x} is calculated as described above (line 1). Next, a segment $s' \in C \cup \{v_x\}$ is selected as a candidate for eviction. However, segments are only considered for eviction if they do not belong to a video for which a session was accepted (line 4). The segment s' is selected as the segment with the maximal earliest reuse time: $s' = \arg\max_{v_i' \in C \cup \{v_x\}} e_{v_i'}$ (lines 5-7). When multiple such segments exist, the LRU order is used as a tiebreaker (lines 8-11). When the evicted segment s' is a cached segment ($s' \in C$), it is removed from the cache and replaced by the new segment v_x (lines 16-18).

Algorithm 2 Outline of the eviction strategy of the election-based caching approach on arrival of a new segment v_x.

```
1: Calculate e_{v_x}
2: s' ← v_x
3: for v_i' ∈ C do
4:     if ∄ start time t_{v'} ∈ Acc for video v' then
5:         Calculate e_{v_i'}
6:         if e_{v_i'} > e_{s'} then
7:             s' ← v_i'
8:         else if e_{v_i'} = e_{s'} then
9:             Calculate LRU ranks LRU_{v_i'} and LRU_{s'}
10:            if LRU_{v_i'} > LRU_{s'} then
11:                s' ← v_i'
12:            end if
13:        end if
14:    end if
15: end for
16: if s' ≠ v_x then
17:    Remove s' from C
18:    Add v_x to C
19: end if
```

The rationale behind this approach is again to keep the segments in the cache that will be needed in the nearest future. However, by not considering segments for removal when a session for that video was accepted, we ensure that the cache will always keep the segments that it intended to store when the session was accepted. In contrast to the threshold-based caching strategy, the election-based strategy provides coordination between the caches, even when no sessions are announced.

V. Scenario Description

To evaluate the proposed approaches, a VoD request trace has been constructed based both on a request trace of the VoD service of a leading European telecom operator and on statistics provided by Conviva[2]. In October 2015, Conviva opened access to viewer experience data, based on the analysis of 4 billion video streams per month spread across 180 countries[3]. The dataset contains statistics about the last year, ranging from Q4 2014 to Q3 2015. Section V-A describes the main characteristics of the constructed request trace, while the applied session duration model, simulating the session interruption behavior, is described in Section V-B. Finally, the considered network topologies are described in Section V-C.

A. VoD trace characteristics

1) Content type: A wide variety of streaming services is monitored by Conviva, with video content consisting of 4 categories: live videos, short videos ($\leq$ 15 minutes), episodic content and movies. As this paper focuses on a VoD scenario, only episodic content and movies are considered. In the remainder of this paper, we will use the terms episodic content and series interchangeably. According to the Conviva dataset, over the last year, 82.28% of the monitored streaming sessions was for episodic content while 17.72% was for movies.

However, while the Conviva dataset contains information about the streaming sessions, it has no statistics about the content catalog. To identify a realistic ratio between series and movies in a VoD content catalog, the Netflix catalog in the USA has been analyzed[4]. At the moment of the analysis, the USA Netflix catalog contained about 7720 movies and 2920 seasons of series.

2) Content characteristics: According to the Conviva dataset, the average bitrate amounts to 2.53Mbps and 1.92Mbps for movies and episodic content respectively. As no information about the catalog is available, the average video duration and the number of episodes per series season were set based on natural values. The duration of a movie was set to be between 60min and 150min while an average series episode has a duration of 20min to 50min. All videos are considered to have a segment duration of 1s. Each season of a series consists of 10 to 30 episodes and the popularity of a series is uniformly distributed across the episodes.

[2]Conviva - http://www.conviva.com
[3]Conviva dataset - http://www.conviva.com/industry-data-portal/
[4]iStreamguide - http://usa.istreamguide.com

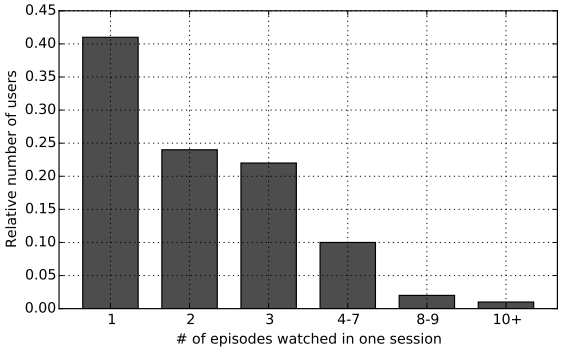


Fig. 3. Distribution of the duration of binge watching sessions as reported by Conviva [2].

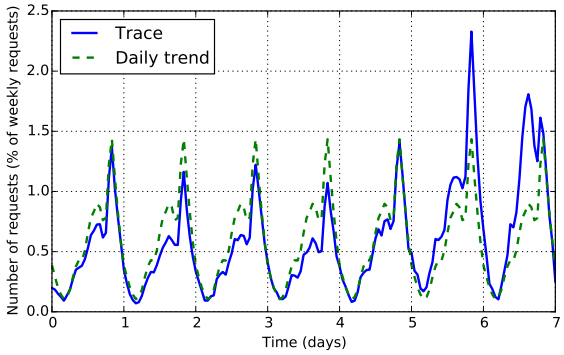


Fig. 4. Relative distribution of the weekly requests starting from Monday 00h00min (day 0) up to Sunday 23h59min (day 7).

3) Binge watching behavior: In August 2015, Conviva released the results of a survey on binge watching [2]. Figure 3 shows the distribution of binge watching durations as published in this report. It can be seen that on average, 2.32 consecutive episodes of a series are watched in a single sitting. The binge watching session can start with any episode of the series.

4) Global content popularity: Given the long-tailed nature of content popularity, the global popularity distribution is commonly represented by the Zipf law [36]. When the content catalog consists of N videos and the videos are ranked according to decreasing global popularity, the relative number of requests r_n for the video with rank n is calculated according to (5), where μ is the scaling-law coefficient of the Zipf distribution. Based on an analysis of a request trace of the VoD service of a leading European telecom operator, in this work, the value of μ was set to 0.9 for both movies and episodic content.

$$r_n = \frac{n^{-\mu}}{\sum_{k=1}^{N} k^{-\mu}} \tag{5}$$

5) Geographical distribution of requests: While the Zipf law models the global popularity distribution, it does not account for the geographical distribution of requests. In this work, the requests are considered to be distributed geographically according to the model proposed by *Tuncer et al.* [37]. When the content catalog consists of N videos and requests originate from L locations, the video with rank n is requested from l_n locations, as defined in (6). In this equation, σ is a strictly positive parameter that defines the characteristics of the distribution. Again based on an analysis of a request trace of the VoD service of a leading European telecom operator, the value of σ was identified to be 2.43.

$$l_n = 1 + (L-1) \times \left(\frac{N-n}{N-1}\right)^{\sigma} \tag{6}$$

6) Request pattern: In order to distribute the video requests over time, we analyzed a request trace of the VoD service of a leading European telecom operator, collected between Saturday February 6, 2010 and Sunday March 7, 2010. Based on this analysis, the weekly request pattern shown in Figure 4 has been extracted. This graph shows the distribution of requests during the week, starting in Monday 00h00min (day 0) up to Sunday 23h59min (day 7), and the average daily trend.

TABLE I
SUMMARY OF RESULTING VOD TRACE CHARACTERISTICS.

Characteristic	**Movies**	**Series**
Units	700	265
Episodes/unit	1	10-30
Rel. nr. of requests	17.72%	82.28%
Avg. bitrate	2.53Mbps	1.92Mbps
Duration	60-150min	20-50min
Avg. binge watching duration	1	2.32
Popularity distribution	Zipf($\mu = 0.9$)	Zipf($\mu = 0.9$)
Geographical distribution	$l_n(\sigma = 2.43)$	$l_n(\sigma = 2.43)$

Using the above characteristics, a VoD request trace of 7 days has been generated, consisting of 125,000 requests spread across 12 locations. The main characteristics of the resulting VoD request trace are summarized in Table I. The total content catalog size amounts to about 4.25Tbyte.

B. Session duration

In contrast to most other related work, we take into account the fact that users do not necessarily stream a video entirely, but may interrupt the session mid-stream. Multiple models have been presented in literature to represent the session duration in video streaming services. *Ullah et al.* provide a survey of user behavior measurements in several types of video streaming services [38]. According to this survey, the session duration of most VoD services can be modeled using an exponential distribution. To deal with the fixed content length in this work, a normalized exponential distribution was used to model the session duration in our experiments. The cumulative probability $p(x)$ of a session to last at most a relative part $x \in [0;1]$ of the video is calculated as shown in (7), where the value of λ depends on the video type.

$$p(x) = \frac{1 - e^{-\lambda x}}{1 - e^{-\lambda}} \tag{7}$$

For movies, this session duration model is applied to every streaming session. On the contrary, given the binge watching behavior for episodic content, the duration model is only applied to the last episode of a binge watching session. The previous episodes in the session are considered to be watched

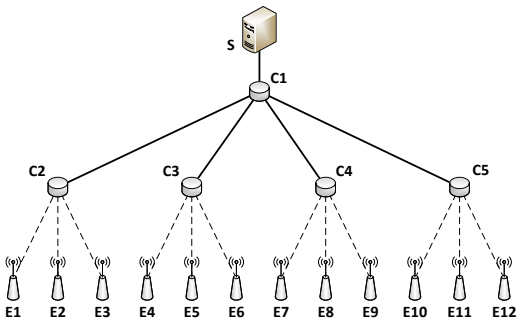


Fig. 5. Evaluated tree topology.

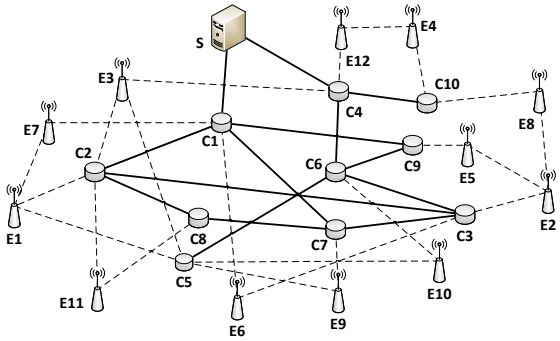


Fig. 6. Evaluated GÉANT-based topology.

completely. According to Conviva, the average session duration amounts 52.23% and 68.43% of the total video length for movies and series respectively. Taking into account the binge watching characteristics, only 26.75% of the last episode of a binge watching session is watched on average. This results in the parameter of the normalized exponential duration model set to $\lambda = -0.21$ for movies and to $\lambda = 3.34$ for series.

C. Network topology

To evaluate the performance of the proposed algorithm in a distributed scenario, two network topologies with different characteristics have been used. In both topologies, a distinction is made between core nodes and edge nodes. All requests are introduced at the edge nodes, while the core nodes interconnect the entire network. Caching capacity is available both at the core nodes and edge nodes. As described in Section V-A, the employed VoD request trace contains 12 cities, which we map onto 12 edge nodes in both topologies.

The first topology represents a three-level tree topology, consisting of 12 edge nodes, 5 core nodes and 1 server node S, hosting all video content. The caching capacity at core nodes $C2$-$C5$ is twice as high as the capacity at the edge nodes, while the capacity of the root core node $C1$ is twice as high as the capacity of the other core nodes. The resulting topology is shown in Figure 5.

The second topology represents a general network, based on the GÉANT topology[5], consisting of 12 edge nodes, 10 core nodes and 1 server node S, hosting all video content. The caching capacity at the core nodes is twice as high as the capacity at the edge nodes. The resulting topology is shown in Figure 6.

In each experiment, the total caching capacity in the network is expressed relative to the total video catalog size. The capacity is spread uniformly across the network, taking into account the capacity ratios described above.

VI. Evaluation results

To characterize the performance of the proposed approaches, multiple performance indicators are evaluated:

- **Hit ratio:** the relative amount of segment requests that could be served from within the Internet Service Provider (ISP) network (in %).

[5]GÉANT Project - http://www.geant.net

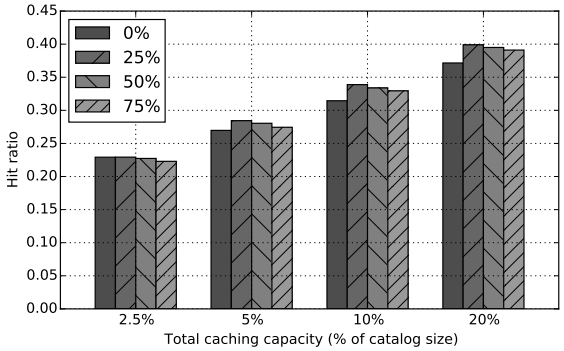


Fig. 7. Impact of the session acceptance threshold α on the hit ratio of the threshold-based caching strategy in the tree topology when all announcements are made without delay.

- **Average bandwidth usage:** the average bandwidth usage in the entire ISP network (in Mbit/s), introduced by the video streaming sessions.
- **Average hop count:** the average number of links a segment crosses between its storage location and the requesting client. This metric indicates how close the relevant content is stored to the end-users.

In all of the evaluations, the performance of the proposed approaches is compared to both the LRU caching strategy and a purely reuse time-based strategy without session announcements and cache coordination, comparable to the approach proposed by *Wu et al.* [18].

A. Influence of the session acceptance threshold α

As described in Section IV-A, the session acceptance threshold α is used in the threshold-based caching approach to decide whether to accept an announced session locally or to cascade it to the next hop. To evaluate the impact of this parameter, the scenario where all announcements are made without delay (i.e., $\beta = 0.0$) is considered first.

Figure 7 shows the influence of the session acceptance threshold α on the hit ratio of the proposed threshold-based caching strategy for various amounts of caching capacity in the tree topology. It can be seen that the best results can be achieved with a session acceptance threshold α of 25%. Similar results are obtained for the general topology and for the other performance indicators.

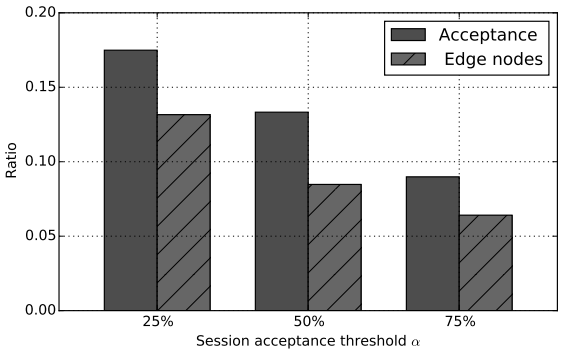


Fig. 8. Impact of the session acceptance threshold α on the session acceptance in the network for the threshold-based caching strategy in the tree topology when all announcements are made without delay and with a total capacity of 5% of the catalog size.

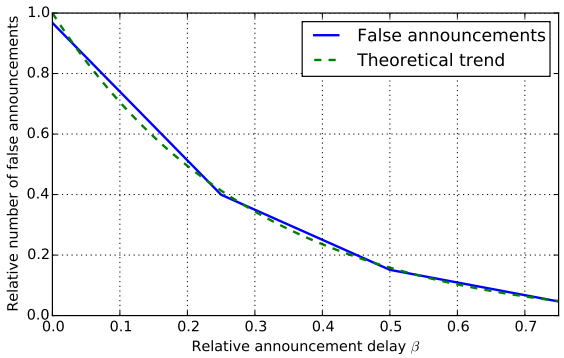


Fig. 9. Impact of the relative announcement delay β on the relative number of false announcements and its theoretical trend.

However, it can be seen that for smaller amounts of caching capacity, the performance of the approach with acceptance threshold $\alpha = 0\%$ comes closer to the performance with acceptance threshold $\alpha = 25\%$. This behavior can be explained by the semantics of the acceptance threshold. With an acceptance threshold of 0%, all sessions are accepted at the edge nodes, without cascading information in the network. This clearly is a suboptimal situation. However, when for example the acceptance threshold amounts 25%, sessions will only be accepted when at least 25% of the segments of that session are expected to be present in the cache. For small caches, chances strongly decrease that a significant amount of segments will be cached at once when multiple sessions are active in parallel. Therefore, only very few sessions will be accepted inside the network and most of the announcement information will be unused. In this case, better results are achieved by accepting all sessions at the network edge, rather than not accepting them at all.

The impact of the threshold α on the session acceptance in the network is shown in Figure 8 for a caching capacity of 5% of the total content catalog in the tree topology. The results for $\alpha = 0\%$ are omitted for visibility reasons, as in this case all sessions are accepted at the edge nodes. As expected, the number of accepted sessions decreases with higher values of α. Furthermore, when α increases, a bigger part of the accepted sessions was accepted in the network core nodes. This can be explained by the fact that core nodes simultaneously perceive requests of multiple edge nodes, such that cached segments can be reused across geographical locations. Similar results are obtained for the general topology.

Given the above analysis, the session acceptance threshold of $\alpha = 25\%$ can generally be selected as the best configuration for the threshold-based caching strategy.

B. *Influence of the relative announcement delay β*

As introduced in Section III, the relative announcement delay parameter β defines at what time in a streaming session of a series episode, the next episode will be announced. Naturally, the value of β serves as a tradeoff. When announcements are made in an early stage (low values of β), the caching network is given a larger future request window, based on which more accurate decisions can be made. On the other hand, when delaying the announcements (larger values of β), false announcements are avoided given that some streaming sessions will have ended before an announcement was made.

The generated VoD trace contains 44091 sittings for series, in total streaming 102627 episodes. In theory, when all announcements are made immediately, 44091 false announcements will be made when announcing the next episode during the last episode of the sitting (neglecting the situation when the last episode of a season is being watched and no future announcement can be made). Figure 9 shows the number of false announcements, relative to the maximal number of false announcements, for the relative announcement delay $\beta \in \{0.0, 0.25, 0.50, 0.75\}$. As expected, the number of false announcements decreases with higher values of β. Furthermore, the number of false announcements clearly follows the theoretical trend that can be calculated based on the session duration model. More specifically, the theoretical number of false announcements for a given value of β is the number of streaming sessions that have not finished after a relative part β of its total duration.

The influence of the relative announcement delay parameter β on the performance of the proposed caching strategies in terms of hit ratio is shown in Figure 10, using the tree topology and a total caching capacity of 5% of the total catalog size. It can be observed that while the number of false announcements significantly decreases with increasing values of β, better performance is achieved for lower values of β for both the threshold-based and the election-based caching strategies. As shown in Table II, the same influence can be perceived on the average hop count and the average total bandwidth usage. Similar results are obtained in the general topology and for different cache sizes. As the LRU and the reuse time-based strategies do not take into account session announcements, their performance is not influenced by the value of β.

Based on the above analysis, it can be concluded that in the considered scenario, the benefits of the earlier availability of future request information exceed the negative impact of the false announcements for both of the proposed caching strategies. Therefore, the clients are configured to immediately announce an episode of a series once the previous episode is

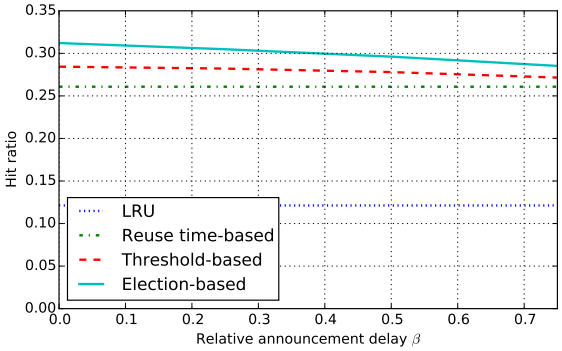


Fig. 10. Impact of the relative announcement delay β on the hit ratio of the proposed caching strategies in the tree topology with a total caching capacity of 5% of the catalog size.

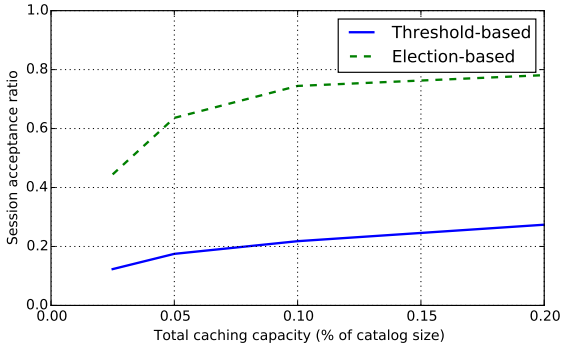


Fig. 11. Session acceptance ratio for both of the proposed caching strategies in the tree topology for different amounts of caching capacity.

TABLE II
IMPACT OF THE RELATIVE ANNOUNCEMENT DELAY β ON THE AVERAGE HOP COUNT AND THE AVERAGE TOTAL BANDWIDTH USAGE IN THE TREE TOPOLOGY WITH A TOTAL CACHING CAPACITY OF 5% OF THE CATALOG SIZE.

	Threshold-based		Election-based	
β	Hop count	Bandwidth	Hop count	Bandwidth
0	2.48	2057Mbps	2.41	2002Mbps
0.25	2.48	2061Mbps	2.43	2016Mbps
0.5	2.49	2068Mbps	2.45	2034Mbps
0.75	2.51	2077Mbps	2.48	2055Mbps

started, i.e., $\beta = 0.0$.

C. Distribution of accepted sessions

The cooperation strategy within the caching approach defines how sessions are accepted in the caching network. Figure 11 shows the relative amount of sessions that were accepted in the tree topology for different amounts of caching capacity. As in the threshold-based approach only announced session can be accepted, the acceptance ratio is calculated based on the number of announcements. In contrast, in the election-based strategy, session can be accepted, independent of whether or not they were announced. Therefore, the acceptance ratio is based on the number of announcements and the number of unannounced sessions. As expected, an increasing trend can be observed with the amount of caching capacity for both approaches. However, for the election-based strategy, significantly more sessions are accepted. This difference can be explained by the nature of the communication in both approaches. For the threshold based approach, sessions will only be accepted when at least a local hit ratio of 25% can be expected. In contrast, in the election-based strategy, sessions can be accepted as soon as a hit can be expected for at least 1 segment of the video.

The cooperation strategy not only influences the session acceptance ratio, but also impacts the distribution of the accepted session across the network. Figure 12 shows the distribution of the accepted sessions for the threshold-based caching strategy across the edge nodes (nodes $E1 - E12$), the core nodes $C2 - C5$ and the root node $C1$ of the tree topology. It can be seen that for small amounts of caching capacity, the majority of the accepted sessions are accepted deep in the tree network.

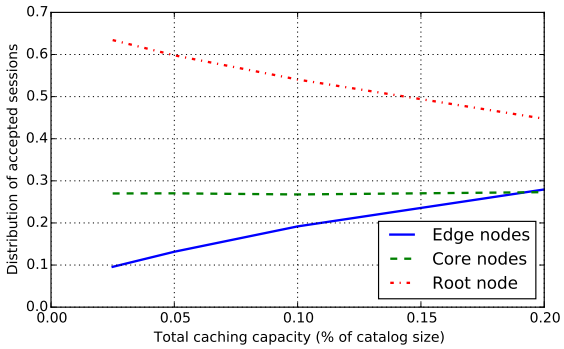


Fig. 12. Distribution of accepted sessions in the tree topology for the threshold-based caching strategy.

Given the threshold of $\alpha = 25\%$, most announcements are cascaded in the edge nodes. In the core nodes, however, caching space can be shared between sessions streaming the same content from different geographical locations. In this way, more sessions can reach the acceptance threshold in the core nodes and at the root node. When the capacity increases, more sessions are accepted directly at the edge nodes as the number of segments from a single video that can be cached grows. Relatively less announcements are cascaded all the way to the root of the tree network, decreasing the relative amount of sessions that are accepted there. However, most of the accepted sessions are still accepted at the core nodes or at the root of the tree.

The distribution of accepted sessions in the tree network for the election-based caching strategy is shown in Figure 13. Compared to the threshold-based strategy, accepted sessions are more evenly distributed across the network for small amounts of caching capacity. When the caching capacity increases, more locally popular content can be cached in the edge nodes, increasing the relative amount of accepted sessions at the edge nodes. However, when the caching capacity increases, relatively more sessions are accepted in the root of the tree network as well. All sessions cross the root $C1$ of the tree network, which thus has knowledge about all active requests in the network. Caching globally popular content in the root of the network can increase local hits for sessions originating from multiple locations. Therefore, the

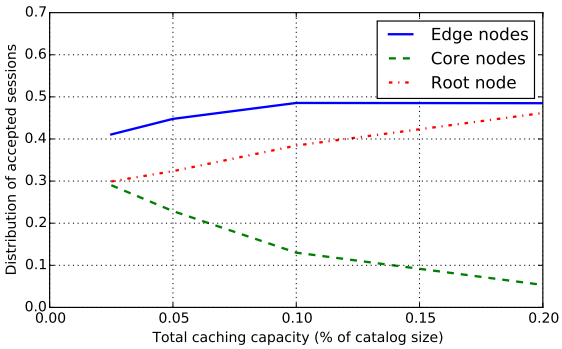


Fig. 13. Distribution of accepted sessions in the tree topology for the election-based caching strategy.

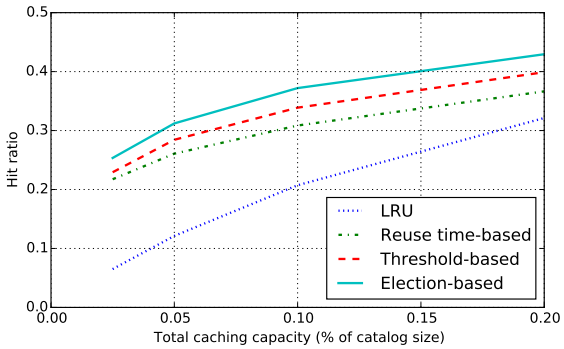


Fig. 14. Performance comparison in terms of hit ratio in the tree topology.

expected gain for globally popular content is the highest in the root node. As the capacity increases, the estimated hit loss decreases, resulting in more sessions accepted in the root node.

D. Performance comparison

Finally, the performance of the proposed approaches can be compared to the LRU approach and a purely reuse time-based approach. For the threshold-based approach, based on the analysis in Section VI-A, the session acceptance threshold was set to $\alpha = 25\%$. The client messaging behavior was configured to immediately send out announcements for subsequent episodes ($\beta = 0.0$), based on the results of Section VI-B.

1) Tree topology: Figure 14 provides a comparison in terms of hit ratio between the proposed approaches and the reference approaches in the tree topology. It can be seen that both proposed approaches significantly outperform the state-of-the-art for each of the evaluated caching capacities. However, the performance gain compared to the reuse time-based approach increases for larger capacities. When more caching capacity is available, the proposed strategies can benefit more from the available session announcements. However, it can be seen that the performance increase compared to the LRU approach decreases for larger capacities, even for the purely reuse time-based approach. This demonstrates that the advantage of taking into account the temporal structure in the video is stronger for smaller caches. Comparing the threshold-based approach to the election-based approach shows that the advanced level of coordination in the caching network yields a constant performance increase, independent of the leased capacity.

The performance in terms of hit ratio, average hop count and average total bandwidth usage inside the tree network is summarized in Table III. It can be seen that when the caching capacity amounts to 5% of the content catalog, using the election-based coordination can relatively increase the hit ratio with 10% while using 3% less bandwidth by bringing the content 3% closer to the end user on average, compared to the threshold-based coordination. The proposed approach results in a hit ratio increase of 20% compared to the state-of-the-art while using 4% less bandwidth. Compared to the LRU approach, the performance increase amounts to 157%, 14% and 14% in terms of hit ratio, average hop count and average total bandwidth usage respectively.

TABLE III
PERFORMANCE COMPARISON IN THE TREE TOPOLOGY.

	Approach	Hit ratio (%)	Hop count	Bandwidth (Mbps)
2.5%	LRU	6.51	2.89	2415
	Reuse time-based	21.74	2.61	2161
	Threshold-based	22.93	2.58	2145
	Election-based	25.35	2.52	2092
5%	LRU	12.13	2.79	2335
	Reuse time-based	26.08	2.53	2092
	Threshold-based	28.44	2.48	2057
	Election-based	31.21	2.41	2002
10%	LRU	20.72	2.62	2197
	Reuse time-based	30.85	2.43	2007
	Threshold-based	33.88	2.36	1957
	Election-based	37.20	2.30	1905
20%	LRU	32.13	2.36	1981
	Reuse time-based	36.65	2.28	1887
	Threshold-based	39.90	2.21	1827
	Election-based	42.95	2.16	1790

2) General topology: As can be seen in Figure 15, similar results can be obtained in the general GÉANT-based topology as in the tree topology, showing the general applicability of our proposed approach. However, it can be seen that each of the evaluated approaches yield better performance in the tree topology. This can be addressed to the hierarchical structure in the tree topology. As each node in the tree has an aggregated view of the streaming sessions in its child nodes, the same caching capacity can be used more efficiently compared to the general topology, lacking any hierarchical structure. Furthermore, the fact that the caching capacity is distributed across a higher number of nodes in the GÉANT-based topology compared to the tree topology results in lower absolute cache sizes, negatively influencing the hit ratio for all of the considered approaches.

Table IV shows the performance in terms of the different performance metric for the general topology. It can be seen that in a scenario where the total caching capacity amounts to 5% of the content catalog, the election-based approach outperforms the threshold-based approach with 13%, 2% and 2% in terms of hit ratio, average hop count and average total bandwidth usage respectively. Compared to the state-of-the-art, the proposed approach results in a hit ratio increase of 22% while using 3% less bandwidth by bringing the content 4% closer to the end user on average.

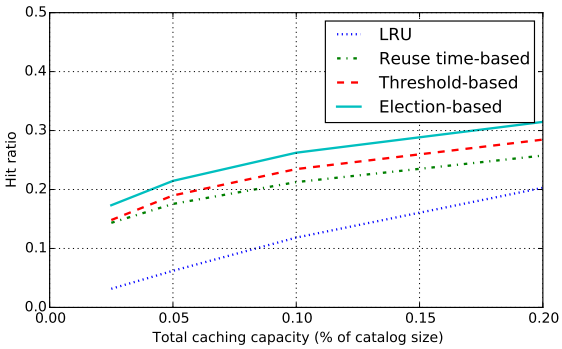


Fig. 15. Performance comparison in terms of hit ratio in the GÉANT-based topology.

TABLE IV
PERFORMANCE COMPARISON IN THE GENERAL TOPOLOGY.

	Approach	Hit ratio (%)	Hop count	Bandwidth (Mbps)
2.5%	LRU	3.16	2.70	2251
	Reuse time-based	14.35	2.49	2067
	Threshold-based	14.82	2.48	2060
	Election-based	17.33	2.42	2016
5%	LRU	6.21	2.64	2204
	Reuse time-based	17.55	2.44	2017
	Threshold-based	18.99	2.41	1996
	Election-based	21.46	2.35	1953
10%	LRU	11.84	2.53	2111
	Reuse time-based	21.26	2.36	1952
	Threshold-based	23.47	2.31	1919
	Election-based	26.24	2.26	1879
20%	LRU	20.29	2.34	1956
	Reuse time-based	25.76	2.25	1862
	Threshold-based	28.46	2.19	1817
	Election-based	31.47	2.15	1784

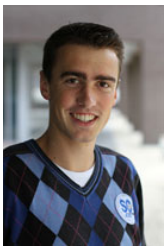
VII. Conclusion

In this paper, two cooperative announcement-based caching strategies for Video-on-Demand (VoD) systems were presented. These strategies not only take into account the temporal structure of videos, but also take advantage of the phenomenon of binge watching where users watch multiple consecutive episodes of the same TV show. This allows a significant part of the streaming sessions to be announced in advance of the actual playout. While both of the proposed approaches apply a reuse time-based replacement strategy, they fundamentally differ in the coordination process inside the caching network. Both approaches have been thoroughly evaluated using a VoD request trace in two network topologies with different characteristics and with different amounts of caching capacity, showing the general applicability of the proposed approach. It was shown that the proposed approach, applying an election-based coordination strategy, can outperform the state-of-the-art with more than 20% in terms of cache hit ratio in a realistic scenario, while simultaneously using 4% less network bandwidth.

References

[1] L. Bélády, "A study of replacement algorithms for a virtual-storage computer," *IBM Systems Journal*, vol. 5, no. 2, pp. 78–101, 1966.
[2] Conviva, "Binge watching: the new currency of video economics," Conviva, Tech. Rep., 2015.
[3] M. Claeys, N. Bouten, D. De Vleeschauwer, W. Van Leekwijck, S. Latré, and F. De Turck, "An announcement-based caching approach for video-on-demand streaming," in *Proceedings of the International Conference on Network and Service Management (CNSM)*, 2015, pp. 1–8.
[4] N. Megiddo and D. S. Modha, "Outperforming LRU with an adaptive replacement cache algorithm," *Computer*, vol. 37, no. 4, pp. 58–65, 2004.
[5] N. Färber, S. Döhla, and J. Issing, "Adaptive progressive download based on the MPEG-4 file format," *Journal of Zhejiang University SCIENCE A*, vol. 7, no. 1, pp. 106–111, 2006.
[6] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in *Proceedings of the International Conference on World Wide Web*, 2001, pp. 36–44.
[7] ——, "Segmentation of multimedia streams for proxy caching," *IEEE Transactions on Multimedia*, vol. 6, no. 5, pp. 770–780, 2004.
[8] S. Chen, B. Shen, S. Wee, and X. Zhang, "Segment-based streaming media proxy: modeling and optimization," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 243–256, 2006.
[9] ——, "SProxy: A caching infrastructure to support internet streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 5, pp. 1062–1072, 2007.
[10] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou, and F. De Turck, "Proactive multi-tenant cache management for virtualized ISP networks," in *Proceedings of the International Conference on Network and Service Management (CNSM)*, 2014, pp. 82–90.
[11] K.-C. Liang and H.-F. Yu, "Adjustable two-tier cache for IPTV based on segmented streaming," *International Journal of Digital Multimedia Broadcasting*, vol. 2012, no. 1, pp. 1–8, 2012.
[12] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six, "Co-operative proxy caching algorithms for time-shifted IPTV services," in *Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications*, 2006, pp. 379–386.
[13] D. De Vleeschauwer and K. Laevens, "Performance of caching algorithms for IPTV on-demand services," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 491–501, 2009.
[14] D. Marinca, A. Hamieh, D. Barth, K. Khawam, D. De Vleeschauwer, and Y. Lelouedec, "Cache management using temporal pattern based solicitation for content delivery," in *Proceedings of the International Conference on Telecommunications (ICT)*, 2012, pp. 1–6.
[15] Z. Xu, X. Guo, Y. Pang, and Z. Wang, "The transmitted strategy of proxy cache based on segmented video," in *Proceedings of the IFIP International Conference on Network and Parallel Computing (NPC)*, 2004, pp. 502–507.
[16] D. Hong, D. De Vleeschauwer, and F. Baccelli, "A chunk-based caching algorithm for streaming video," in *Proceedings of the Workshop on Network Control and Optimization*, 2010, pp. 1–8.
[17] B. Van Roy, "A short proof of optimality for the MIN cache replacement algorithm," *Information Processing Letters*, vol. 102, no. 2, pp. 72–73, 2007.
[18] T. Wu, K. De Schepper, W. Van Leekwijck, and D. De Vleeschauwer, "Reuse time based caching policy for video streaming," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC)*, 2012, pp. 89–93.
[19] K. De Schepper, B. De Vleeschauwer, C. Hawinkel, W. Van Leekwijck, J. Famaey, W. Van de Meerssche, and F. De Turck, "Shared Content Addressing Protocol (SCAP): Optimizing multimedia content distribution at the transport layer," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, 2012, pp. 302–310.
[20] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.
[21] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the ICN workshop on information-centric networki*, 2012, pp. 55–60.
[22] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, vol. 3, no. 1, 2001, pp. 1416–1424.
[23] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, no. 7, pp. 1305–1314, 2002.
[24] X. Tang and S. T. Chanson, "Coordinated en-route web caching," *IEEE Transactions on Computers*, vol. 51, no. 6, pp. 595–607, 2002.
[25] A. Jiang and J. Bruck, "Optimal content placement for en-route web caching," in *Proceedings of the IEEE International Symposium on Network Computing and Applications*, 2003, pp. 9–16.

[26] K. Poularakis and L. Tassiulas, "Optimal cooperative content placement algorithms in hierarchical cache topologies," in *Proceedings of the Annual Conference on Information Sciences and Systems (CISS)*, 2012, pp. 1–6.
[27] J. Ni and D. H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Communications Magazine*, vol. 43, no. 5, pp. 98–105, 2005.
[28] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 1–9.
[29] J. Zhang, "A literature survey of cooperative caching in content distribution networks," *Computing Research Repository*, vol. 1210, no. 71, pp. 1–5, 2012.
[30] Z. Li and G. Simon, "In a Telco-CDN, pushing content makes sense," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 10, no. 3, pp. 300–311, 2013.
[31] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, 2013.
[32] Z. Li and G. Simon, "Time-shifted TV in content centric networks: The case for cooperative in-network caching," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2011, pp. 1550–3607.
[33] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM) Workshops*, 2012, pp. 268–273.
[34] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 10, no. 3, pp. 286–299, 2013.
[35] Y. Li, H. Xie, Y. Wen, C.-Y. Chow, and Z.-L. Zhang, "How much to coordinate? Optimizing in-network caching in content-centric networks," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 12, no. 3, pp. 420–434, 2015.
[36] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.
[37] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou, "More control over network resources: an ISP caching perspective," in *Proceedings of the International Conference on Network and Service Management (CNSM)*, 2013, pp. 26–33.
[38] I. Ullah, G. Doyen, G. Bonnet, and D. Gaïti, "A survey and synthesis of user behavior measurements in P2P streaming systems," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 3, pp. 734–749, 2012.



Maxim Claeys obtained a masters degree in computer science from Ghent University, Belgium, in June 2012. In August 2012, he joined the Department of Information Technology at Ghent University, where he is active as a Ph.D. student. His main research interests are the application of autonomic network management approaches in multimedia delivery. The focus of this research is mainly on the end-to-end Quality of Experience optimization, ranging from the design of autonomous clients to intelligent in-network decision taking.

Niels Bouten obtained a masters degree in computer science from Ghent University, Belgium, in June 2011. In August 2011, he joined the Department of Information Technology at Ghent University, where he is active as a Ph.D. student. His main research interests are the application of autonomic network management approaches in multimedia delivery. The focus of this research is mainly on the end-to-end Quality of Experience optimization, ranging from the design of a single autonomic control loop to the federated management of these distributed loops.

Danny De Vleeschauwer received the degree of electrical engineer and the Ph.D. degree in applied sciences form Ghent University, Belgium, in 1985 and 1993 respectively. In 1986 he joined the Telecommunications and Information processing department of Ghent University, Belgium, where he worked first on image processing and then on the applications of queuing theory in packet-based networks. Since 1998 he is working for Alcatel-Lucent, Antwerp, Belgium, currently in Bell Labs CTO, on performance issues related to packetized transport of multimedia and data services. He is guest professor in the Stochastic Modeling and Analysis of Communications Systems research group of the University of Gent. His main research interests include signal processing, coding technology, queuing theory and their application in the multimedia transport over packet-based networks.

Werner Van Leekwijck obtained a Master of Science degree in Electrical Engineering from the University of Leuven, Belgium, in 1990, and an additional Master degree in Knowledge and Information Technology from the University of Ghent, Belgium. He joined Alcatel in 1990 and he has held various technical and management positions in research and product development in the area of IP multimedia technology and networks. Currently, he is a senior research manager in Bell Labs in Antwerp, and is responsible for future multimedia delivery architectures.

Steven Latré is an assistant professor at the University of Antwerp and iMinds, Belgium. He received a Master of Science degree in computer science from Ghent University, Belgium and a Ph.D. in Computer Science Engineering from the same university with the title Autonomic Quality of Experience Management of Multimedia Services. His research expertise focuses on management and control of both networking and computing applications. In this context, he has focused on Quality of Experience optimization and management, multimedia management, software defined networking and network virtualization. He currently focuses on the management of large-scale Internet of Things environments and the interplay with cloud computing. He has also been involved in several national and European research projects (e.g., FP6 MUSE, FP7 ECODE, FP7 OCEAN, FP7 FLAMINGO). He is author or co-author of more than 65 papers published in international journals or in the proceedings of international conferences. He is the recipient of the IEEE COMSOC award for best PhD in network and service management 2012, the IEEE NOMS Young Professional award 2014 and is a member of the Young Academy Belgium.

Filip De Turck leads the network and service management research group at the Department of Information Technology of the Ghent University, Belgium and iMinds (Interdisciplinary Research Institute in Flanders). He (co-) authored over 450 peer reviewed papers and his research interests include telecommunication network and service management, and design of efficient virtualized network systems. In this research area, he is involved in several research projects with industry and academia, serves as the vice chair of the IEEE Technical Committee on Network Operations and Management (CNOM), chair of the Future Internet Cluster of the European Commission, and is on the TPC of many network and service management conferences and workshops and serves in the editorial board of several network and service management journals.